⑲日本国特許庁(JP)　⑪特許出願公開

⑫公開特許公報(A)　平1−217536

⑤Int.Cl.⁴	識別記号	庁内整理番号	㊸公開　平成1年(1989)8月31日
G 06 F 15/16	380	Z−6745−5B	
9/44	330	D−8724−5B	
15/16	410	6745−5B	審査請求　未請求　請求項の数 2　(全10頁)

㊹発明の名称　異種言語混在形プログラム処理コントローラ

㉑特　願　昭63−40870
㉒出　願　昭63(1988)2月25日

㉒発明者　大橋　章宏　茨城県日立市大みか町5丁目2番1号　株式会社日立製作所大みか工場内
㉒発明者　山岡　弘昌　茨城県日立市大みか町5丁目2番1号　株式会社日立製作所大みか工場内
㉒発明者　高倉　満郎　茨城県日立市大みか町5丁目2番1号　株式会社日立製作所大みか工場内
㉑出願人　株式会社日立製作所　東京都千代田区神田駿河台4丁目6番地
㉔代理人　弁理士　秋本　正実

明　細　書

1．発明の名称

異種言語混在形プログラム処理コントローラ

2．特許請求の範囲

1．制御用の複数のプロセッサと、制御用プログラム内の各処理を記述している言語に基づき、上記複数のプロセッサの中で最適なプロセッサを判断し、処理を最適なプロセッサに分配する手段と、を有する異種言語混在形プログラム処理コントローラ。

2．制御用の複数のプロセッサと、ジョブコントローラと、該ジョブコントローラとプロセッサ間通信を行うバスとを備えると共に、ジョブコントローラは、

プログラムの各処理を記述している言語種別を認識する手段と、各制御用プロセッサの言語処理能力を認識する手段と、各制御用プロセッサの実装／未実装を認識する手段と、言語種別及び言語処理能力、及び実装の条件のもとで上記複数のプロセッサの中で最適なプロセッサを判断し処理を該プロセッサに分配する手段と、を有してなる異種言語混在形プログラム処理コントローラ。

3．発明の詳細な説明

〔産業上の利用分野〕

本発明は、プロセス制御用コントローラに係り、特に異なる言語により記述された処理が混在するプログラムを処理するプロセス制御用コントローラに関する。

〔従来の技術〕

従来のプロセス制御用コントローラのプログラム言語動向については、電気学会誌昭56−10、第957頁から第964頁の「PCにおける言語の動向」において論じられているように、ハード面では、トランジスタによる専用回路形から汎用マイクロプロセッサ形、ビットスライス形へと発展してきた。また、ソフト面では、この間、ラダーシーケンスからブロック図等のグラフィック言語、さらには、FORTRAN、C言語等の汎用言語による数式演算へと発展してきた。

現在では、プロセス制御用コントローラの世界において、多数のプロセッサと多数のプログラミング言語形態が存在する。

このため、第２図（イ）に示すようにラダー、ブロック夫々で作成したプログラム１５，１７を各言語用のコントローラ１６，１８で実行する場合は問題ない。即ち、コントローラａはラダー表現によるプログラム処理を高速に実行し、コントローラｂはブロック図表現によるプログラム処理を高速に実行する。このため、コンパイラ１１，１２は既存のものである。しかし第２図（ロ）に示すようにラダーで記述されたプログラムをブロック図用のコントローラ１８で実行する場合、専用のコンパイラ１４を作成する必要がある。ブロックからラダーへも同様なコンパイラ１３を必要とする。仮に、プロセッサの種類がｍ、プログラミング言語の数がｎ存在したとすると、ｍ×ｎ個のコンパイラを作成する必要が出てくる。これを避ける方法として、「オートメーション」（３３巻第１号，Ｐ１１～１５）中に記載されている中間言語を用いる方法がある。中間言語の個数をℓ個（ℓ≦ｍ）とすると、コンパイラはℓ×ｍですむことになる。

〔発明が解決しようとする課題〕

しかし、現在の技術では、すべてのプログラミング信号を包含するような中間言語を開発することは困難であり、用途別にいくつかの中間言語の形態を持つのが通常である。

ところが、上記「オートメーション」中の従来技術では、中間言語間の変換が依然として残っており、特にその第１４頁記載の第３図に示すような異なる言語で記述された処理が混在するプログラムの処理については配慮されておらず、従来開発されたソフトウエア財産をファンクションモジュールという形で組み合わせて利用することができず、各言語で記述された処理を共通の中間言語に変換し、さらに使用するプロセッサの言語形態に合わせた変換が必要となるという問題があつた。

本発明の目的は、既存のソフトウエア財産を変更することなく、第３図に示すようにメモリ上に作成された異種言語混在形プログラムを実行可能なプロセス制御用コントローラを提供することにある。

〔課題を解決するための手段〕

本発明は、プロセス制御用コントローラ内に、複数のプロセッサを設け、更に、プログラム内の各処理を記述された言語に基づき処理を行わせる場合に、各言語処理に最適なプロセッサを判断し、当該処理を最適プロセッサに分配する制御手段を設けた。

更に、本発明は、上記制御手段は、ジョブコントローラで実現させた。このジョブコントローラは、複数のプロセッサとバスを介して接続させた。ジョブコントローラはバスを介して各プロセッサと通信を行わせることとし、このバスを介しての通信のもとで、判断し分配のために以下の４つの手段を有する。即ち、ジョブコントローラは、プログラム内の各処理の言語種別を認識する手段、各プロセッサの言語処理能力を認識する手段、各プロセッサの実装状態を認識する手段、この判断結果に従ってプロセッサへの処理を分配指令する手段を持つ。ここで処理とはプログラム実行のことであり、分配指令とはローディングを意味する。

〔作用〕

本発明によれば、制御手段によつて、各言語処理に最適なプロセッサを判断し、このプロセッサへの処理の分配が達成できる。

更に、ジョブコントローラで実現させた例では、当該ジョブコントローラは、プログラム内の各処理の言語種別を認識する手段により、実行する処理の言語を認識し、次に各プロセッサの当該言語に対する処理能力を認識する手段により、最も処理能力の高いプロセッサを選択する。さらに、当該プロセッサが実装されていることを、実装状態を認識する手段により確認した後、分配指令手段により当該プロセッサへ当該処理のプログラムをロードする。

当該プロセッサが実装されていない場合は、再び当該言語に対する各プロセッサの処理能力を認識する手段により、次に処理能力の高いプロセッ

サを選択する。その後再び実装状態を認識する手段により、当該プロセッサが実装されていることを確認し、実装されていれば、分配指令手段により当該処理プログラムを当該プロセッサへロードし、実装されていなければ、再び当該言語に対する各プロセッサの処理能力を認識する手段に戻り、同様の処理を繰り返す。

当該言語に対する各プロセッサの処理能力を認識する手段には、プロセス制御用コントローラに実装される可能性があるすべてのプロセッサが登録されており、かつ、プロセス制御用コントローラにプロセッサが１つも実装されないということは、事実上有り得ないため、当該ジョブコントローラの処理が無限ループに落ちいることはない。

尚、かかる最適プロセッサへ分配された処理の実行は、実際の制御処理そのものである。従つて、この実際の制御処理の中で、順序に従つて各プロセッサが起動され、制御動作を続けてゆく。この際のプログラムの起動は、ジョブコントローラが行わせる。

〔実施例〕

本発明の実施例を図面を用い説明する。

第４図は、本発明のプロセス制御用コントローラを用いたシステム構成の一実施例である。

プロセス監視装置２０と本発明のプロセス制御用コントローラ１００，２００が、ネットワーク１０を介して接続される。また、ＰＩ／０３０，３１がプロセス制御用コントローラ１００へ、ＰＩ／０３２，３３，３４がプロセス制御用コントローラ２００へそれぞれ接続される。

プロセス監視装置２０は、プロセス制御用コントローラ１００，２００に実行させるプログラムをネットワーク１０を介して当該プロセス制御用コントローラ１００，２００のメモリへそれぞれ書き込む。一般にプロセス監視装置は、プロセスの各種状態量や条件の表示部及びプロセス制御用の各種操作を行う操作部より成る。しかし、昨今は、この監視・操作機能の他にプログラム作成支援能力を持たせる例が多い。本実施例では、プロセス監視装置２０に、このプログラム作成支援機能を持たせた。これによつて、オペレータは、プロセス監視装置２０でプログラムを作成する。

更に、監視装置２０は、この作成したプログラムをコンパイルし、オブジエクトプログラムを作成する。

ここで、作成プログラムは、ＰＣ（プログラマブル　コントローラ）の場合、異種言語混在のプログラムであることが多い。例えば、異種言語にはラダー形式、ブロック形式等種々存在する。各言語形式のもとでコンパイルされたオブジエクトプログラムは、それぞれ独自のマシンコード形式となる。複数の制御用プロセッサ１０４～１０６は、それぞれ、ラダー形式用言語処理用とか、ブロック形式用言語処理用とかとされ、１つのプロセッサは１つの言語処理用に機能化されている。従つて、１つのプロセッサが、ラダー形式用言語処理用であつて、ブロック形式言語処理用の機能処理を行わせたい場合には、ブロック形式言語処理用の機能を付加することが必要とする。この機能付加は、ブロック形式言語処理用のインタプリタを当該プロセッサに付加することによつて可能である。

尚、異種言語混在プログラムとは、１つのプログラム中に、一部はラダー形式、他の一部がブロック形式とかで記述されている場合、全体がラダー形式で記述されるかそのサブルーチンがブロック形式で記述されている場合、更には、複数のプログラムが存在し、１つはラダー形式、他の１つはブロック形式とかとなつている場合、のいずれをも指すものとする。

プロセス監視装置によるプログラム作成の有力な方法を以下で述べる。プロセス監視装置にデータベースを接続する。このデータベースには過去に作成されたプログラム（ラダーとかブロック図等）がコンパイルされたオブジエクト化された状態（即ちマシンコード化）で格納しておく。オペレータ（プログラマ）は、このプログラムのメニューが表示されている画面をみて、プログラムとして必要なメニューを選択する。この選択したメニューを組合せてプログラムができ上る。この結

果、形成されたプログラムは、既存の異種言語組合せ形プログラムとなる。このわかりやすい一例を第１４図に示す。第１４図で、表示メニューとして①，②，③が表示され、実際のプログラムは①，②のみとすると、メニュー①，③を選択し、これを組合せる。この組合せ結果をＡとして示した。結果Ａは画面に示すようにしてもよく、オペレータの頭の中に存在するものであってもよい。かくして、プログラムとして①と③とのリングしたプログラムが得られる。ことで、Ｂに示す如く、この①と③との組合せプログラムはラダー，ブロックなる異種言語組合せ形プログラムになる，このプログラムは、データベースからマシン言語化された形で取り出され、ジョブコントローラ１０１のメモリ１０２へ送出されラッチされる。

この他に、データベースを利用しないで監視装置で１つ１つプログラムを作成するやり方もある。この方法も本発明に含まれる。

第１図は、第４図に用いられている本発明のプロセス制御用コントローラ１００（コントローラ２００も考え方は同じ）の構成を示す。

プロセッサ（Ａ）１０４は、ラダー図によるプログラムの処理能力が高いプロセッサ、プロセッサ（Ｂ）１０５はBASIC 言語等の汎用言語によるプログラムの処理能力が高いプロセッサ、プロセッサ（Ｃ）１０６は、ブロック図によるプログラムの処理能力が高いプロセッサである。これらのプロセッサ１０４，１０５，１０６は、内部バス１０８を介して、ジョブコントローラ１０１，メモリ１０２，ネットワークインタフエース１０３，ＰＩ／Ｏインタフエース１０７と接続される。

第４図のプロセス監視装置２０から送信されたプログラムは、ネットワークインタフエース103，内部バス１０８を通つて、メモリ１０２へ書き込まれる。このプログラムは例えばＰＣ処理のプログラムであり、種々の言語形式で記述されたプログラムより成る。

ジョブコントローラ１０１に第５図のジョブ言語種別テーブル、第６図の言語処理性テーブル、第７図のプロセッサ登録テーブルを具備させる。

第４図のプロセス監視装置２０では、プログラムをプロセス制御用コントローラ１００へ送信する際、プログラムと共に、第５図のジョブ言語種別テーブルの内容を同時にプロセス制御用コントローラ１００へ送信し、メモリ１００内にプログラムと共に格納する。ここで、第５図のジョブ言語種別テーブルは、プログラムアドレスの順序に従つて、ローディングするジョブ言語種別、及び処理性テーブル配列番号を登録してある。

第６図の言語処理性テーブルは、配列番号０～３対応に言語処理のためのプロセッサ相互の優先関係を規律する。

第７図のプロセッサ登録テーブルは、バス108への各予想プロセッサＡ，Ｂ，Ｃの接続の有無を規律する。第７図のテーブルでは、プロセッサＡ，Ｂ，Ｃのすべてが接続有りとされている。接続されていなければ、その未接続のプロセッサ対応に「有」の代りに「無」の表示を行う。

尚、第１図で、ＰＩ／Ｏインターフエース107より取り込まれ各プロセッサＡ，Ｂ，Ｃにて処理される際に、内部バス１０８を伝送するデータの形式は同一とする。

尚、第６図の言語処理性テーブルは、言語形式の処理速度の大小の順で優先関係を規定した。例えば、配列０にあつては、プロセッサＡが一番処理速度が速く、次いでプロセッサＢ、プロセッサＣとなる。ここで、各プロセッサは、先に述べたように単一の言語形式処理機能を持ち、その他の形式の言語処理のためにはインタプリタを付加させることとした。従つて、配列０の事例では、プロセッサＢとＣとは、それぞれラダー形式処理可能なインタプリタを取りつけているとみるべきである。同様に、配列１では、プロセッサＣがブロック図形式を処理する機能を持ち、プロセッサＡとＢとは、ブロック図処理用のインタプリタを取りつけているべきとみるべきである。他も同様である。

従つて、他言語処理用のインタプリタを取りつけていない場合には、各配列内の優先関係はなく、配列０はプロセッサＡ指定、配列１はプロセッサ

C指定といつた具合いになる。本発明にとつては、両者共に適用可能である。

次に、第8図のフローに従いジョブコントローラ101の処理を説明する。

ジョブコントローラ101は、まず、第5図のジョブ言語種別テーブルを読み出すことにより、第1番目のジョブがメモリ102のアドレス0000～0100に存在し、ジョブ言語がラダーであることを認識し、さらに、ラダー言語に対する各プロセッサの処理能力を示す言語処理性テーブルの配列番号が0であることを認識する（ステップ50）。

次に、当該配列番号0に基づき、第6図の言語処理性テーブルを読み出すことにより、プロセッサ（A）104がラダー言語の処理能力が最も高いことを認識する（ステップ51）。

さらに、第7図のプロセッサ登録テーブルを読み出し（ステップ52）、プロセッサAが実装されていることを確認する（ステップ53）。

確認後、当該プログラムをメモリ102からプロセッサ（A）104へロードする（ステップ54）。プロセッサAでは、このプログラムを内部メモリ（又はレジスタ）にラッチする。

以下同様に、アドレス0100の「ブロック」につき同様の処理を行う。次いでアドレス0200の「ラダー」、アドレス0300の「FA－BASIC」の処理を行い、終了する。

次に、第9図に示すように制御用コントローラ100にプロセッサ（C）106が実装されていない場合のジョブコントローラ101の処理動作を第8図のフローに従い説明する。この場合、第7図のプロセッサ登録テーブルは、第10図のように変更しておく。

先ず当該ジョブコントローラ101は、第5図のテーブルに従つて第1番目のジョブ（ラダー形式）を第8図のフローに従いプロセッサAへ前述したようにロードした後、第2番目のジョブを第5図のジョブ言語種別テーブルから読み出し、プログラムがメモリのアドレス0100～0200に存在し、ジョブ言語はブロック図、処理性テーブル配列番号が1であることを認識する（ステップ50）。

次に、第6図の言語処理性テーブルの配列番号1の内容より、プロセッサ（C）106が、ブロック図によるプログラムの処理能力の最も高いプロセッサであることを確認する（ステップ51）。

次に、第10図のプロセッサ登録を読み出し（ステップ52）、プロセッサ（C）106が実装されているか否かが点検され、この場合プロセッサCが実装されていないため、再び処理を戻し（ステップ53）、第6図の言語処理性テーブルを読み出し、2番目に処理能力が高いプロセッサは、プロセッサ（B）105であることを認識する（ステップ51）。

次に再び第10図のプロセッサ登録テーブルを読み出し（ステップ52）、プロセッサ（B）105が実装されていることを確認し、（ステップ53）、プロセッサ（B）105へ当該プログラムをロードする（ステップ54）。

本実施例では異種言語の組み合せとして、ラダー図、ブロック図、FA－BASICを示したが、他の言語（例えば、C言語、ディシジョンテーブル等）の組み合せにおいても、同様の効果が得られることは言うまでもない。

以上、本実施例によれば、異なる言語によつて書かれたジョブプログラムを同一のプロセス制御用コントローラ上で実行することが可能となり、従来各プロセス制御用プロセッサ上で開発されてきたプログラムを変更することなく利用することができるため、ソフトウエアの再利用性が良い。

また、プロセッサの実装、未実装に対しても、第7図のプロセッサ登録テーブルを変更するのみでよいため、システム変更に対する柔軟性に富む。

さらに、他の実施例として、第11図に示すようなPI／Oインタフエース107が実装されていない場合も、第4図のネットワーク10上を転送されるデータを第11図のネットワークインターフエース103を介してプロセス制御用コントローラ100内に取り込み、以後、第1図の場合と同様の処理を行う。この際、プロセス制御用コ

ントローラ１００で処理された結果のデータは、ネットワークインタフエース１０３を介して再びネットワーク１０上へ送出される。

また、さらに他の実施例としては、第１２図に示すように、ネットワークインタフエース１０３が実装されていない場合が考えられるが、この場合は、プログラムのローディングをネットワークを介さず、各コントローラ毎に行うことで、スタンドアロンのコントローラとして、第１図の場合と同様の処理を行う。

次に、本実施例での全体像をつかむための、処理系統図を第１３図に示す。先ず、プロセス監視装置を利用してプログラム作成し、併せてこの作成プログラムをコンパイルし、オブジエクトプログラムを得、且つ相互のリンクをはかる。次に、メモリ１０２へ、このプログラム（実行モジユール）をロードする。併せて各種テーブルもメモリ１０２へ格納する。ロードが終了すると、コントローラ１００を起動する。この起動によりジョブコントローラ１０１は、プロセツサＡ，Ｂ，Ｃへ実行モジユールの最適配分を行う。図では、プログラムの処理の順序に従つて、ラダーオブジエクトプログラムをプロセツサＡのメモリへ、次にブロツク図オブジエクトプログラムをプロセツサＣへ、最後にＦＡ－BASIC のオブジエクトプログラムをプロセツサＢへ、ロードし、分配を終了する。

以上の各プロセツサのロード後、ジョブコントローラは、各プロセツサの起動をかける。この起動順序は、プロセツサＡ→Ｃ→Ｂの順である。

種々の適用、変形例を述べる。

(1) 本実施例では、ジョブコントローラが、判断分配の役割を果したが、分散して実現することもできる。また、制御用ＣＰＵの１つにその役割を持たせてもよい。

(2) 制御用コントローラ以外にも、異種言語を取扱うことのできる情報処理用計算機システムについても、適用できる。

(3) プログラム作成支援のもとで作成したプログラムをメモリ１０２に送出することとしたが、テープ等で入力させるようにしてもよい。この場合、コンパイル機能は、ジョブコントローラに持たせてもよい。

〔発明の効果〕

本発明によれば、従来各プロセス制御用プログラム言語対応のプロセツサ上で開発されたソフトウエアを１つのプロセス制御用コントローラで実行することが可能であるため、既存のソフトウエア財産をファンクションモジユールとして組み合わせて新しいプログラムを作成し、プログラムがどのような言語形態で記述されているか、ユーザは意識することなく実行することができる。

また、プロセス制御用コントローラ内のプロセツサの種類、規模の変更時は、ジョブコントローラの各テーブルを変更するのみで対応でき、拡張性、柔軟性に富んだプロセス制御用コントローラを得ることができる。

4．図面の簡単な説明

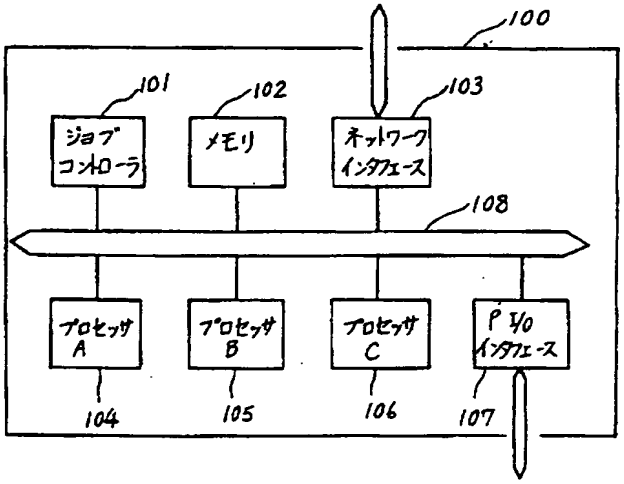
第１図は、本発明のプロセス制御用コントローラの一実施例図、第２図は、従来技術によるプロセス制御用コントローラでのプログラム処理を示す図、第３図は、異種言語混在形プログラムの一例を示す図、第４図は、本発明のプロセス制御用コントローラを使用する際のシステム構成の一実施例図、第５図は、本発明のプロセス制御用コントローラ内にあるジョブコントローラが所有するジョブ言語種別テーブルの一実施例図、第６図は、当該ジョブコントローラが所有する各プロセツサの言語処理性テーブルの一実施例図、第７図は当該プロセス制御コントローラが第４図の構成を採つた場合に当該ジョブコントローラが所有するプロセツサ登録テーブルの一実施例図、第８図は、当該ジョブコントローラが、最適なプロセツサへ各処理のプログラムを分配するときの処理フロー、第９図は、本発明のプロセス制御コントローラの一実施例図、第１０図は、当該プロセス制御コントローラが、第９図の構成を採つた場合のプロセツサ登録テーブルを示す図、第１１図はＰＩ／Ｏインターフエースを実装していない場合の系統図、第１２図はネットワークインタフエースを実装していない場合の系統図、第１３図は全体処理系統

図、第１４図は有効なプログラム作成説明図である。

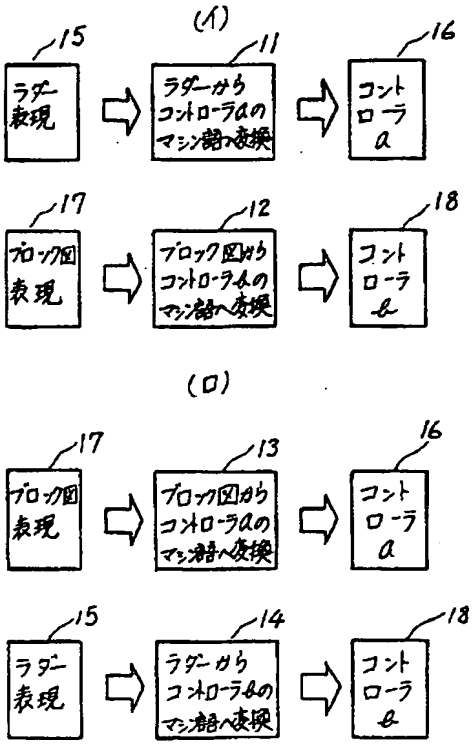
１００…制御用コントローラ、１０１…ジョブコントローラ、１０４，１０５，１０６…プロセッサ。

代理人　弁理士　秋本正実

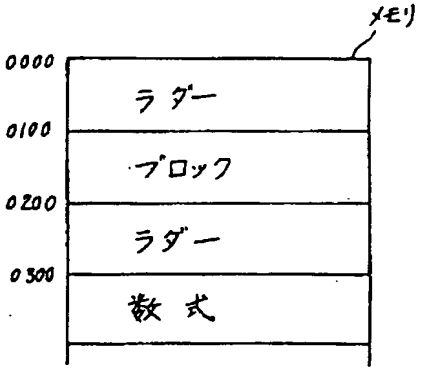
第１図



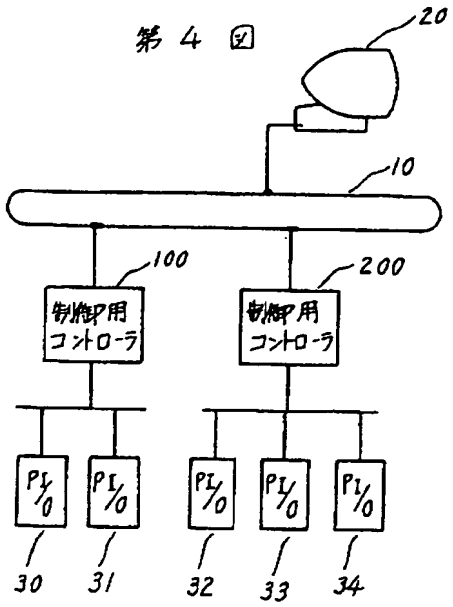
第２図



第３図



第4図



第5図

ジョブ言語種別テーブル

プログラムアドレス	ジョブ言語種別	処理性テーブル配列番号
0000	ラダー	0
0100	ブロック図	1
0200	ラダー	0
0300	FA-BASIC	3

第6図

言語処理性テーブル

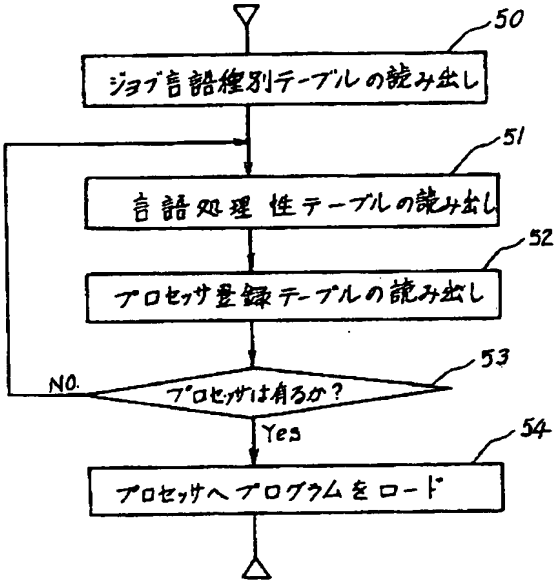
配列	言語処理性
0	プロセッサA > プロセッサB > プロセッサC
1	プロセッサC > プロセッサB > プロセッサA
2	プロセッサB > プロセッサA > プロセッサC
3	プロセッサB > プロセッサC > プロセッサA

第7図

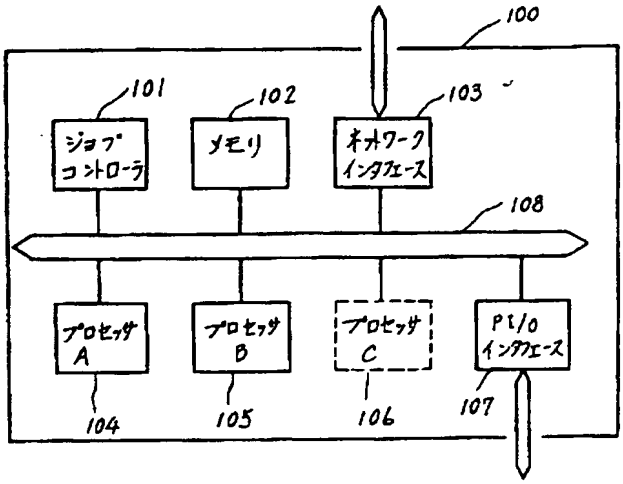
プロセッサ登録テーブル

プロセッサ名	プロセッサA	プロセッサB	プロセッサC
実装状態	有	有	有

第 8 図



第 9 図

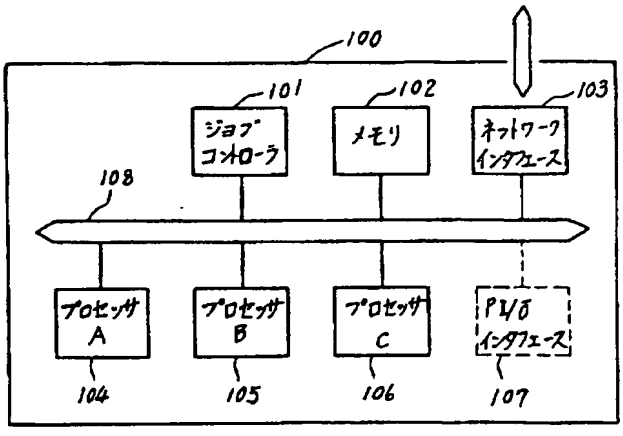


第 10 図

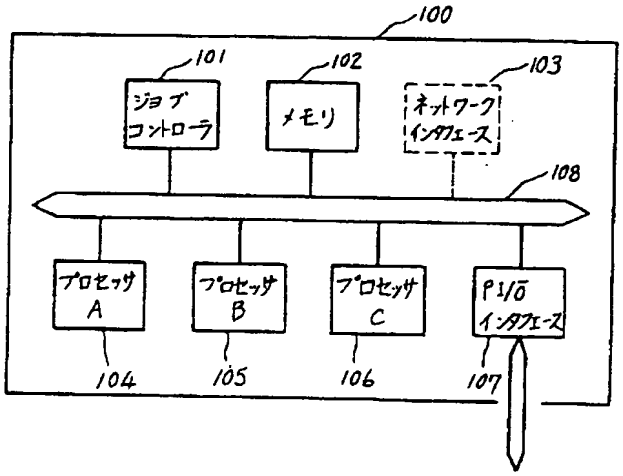
プロセッサ登録テーブル

プロセッサ名	プロセッサA	プロセッサB	プロセッサC
実装状態	有	有	無

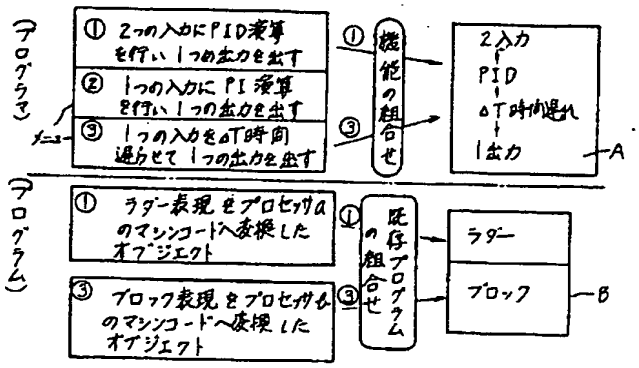
第 11 図



第 12 図



第 14 図



第 13 図